# Towards the Quality Evaluation of Software of Control Systems of Nuclear Power Plants: Theoretical Grounds, Main Trends and Problems

Elena Jharko
*V.A. Trapeznikov Institute of Control Sciences, 65 Profsoyuznaya Street, Moscow, Russia*

Keywords: Quality Assurance, Software, Quality Model, Automated Process Control System (APCS), Nuclear Power Plant (NPP).

Abstract: The paper considers issues of implementing works on the evaluation of the software quality of control systems of nuclear power plants in the part of theoretical grounds, main trends and problems in this branch.

## 1 INTRODUCTION

The process of development of automation of complex technological plants with high operation risk in the power engineering and other branches of industry is characterized by a tendency of development and adoption in the make-up of regular tools of upper level of automated process control systems (APCS) of systems of operator information support (Byvaikov et al., 2006, Poletykin et al., 2006, Jharko, 2008, Jharko and Zaikin, 2011).

In the last decade, automatic process control systems have led to a qualitatively new level of development. Such a level is concerned with an increased level of the automation of control plants and, as a consequence, a growth of the number of control and diagnostic signals processed by the control system per time unit. From another hand side, practically linear growth of the capacity of computer systems that may be used in APCS has enabled one to implement considerably more complex algorithms of control an analysis of data by use of high-performance soft- and hardware tools on computations. However, the qualitative jump in the make-up of solved problems, which has been the case, made one to reconsider the relationship components of the life cycle of the software.

Such changes are clearly traced by use of an example of development of software for NPP APCS with required life time being not less than 30 years. This considerably exceeds the average time of life and storing of hardware, achieved at present, and makes one to pay more attention to careful development of the stage of modification and maintenance of software (SW) developed (Jharko, 2011).

The quality assurance is a continuous process in the course of the whole software life cycle (ISO/IEC 12207:2008), which covers:

- Methods and tools of the analysis, design, and coding;
- Technical reports being implemented at the each step of the software development;
- Procedure of multi-level testing;
- Monitoring the software documentation and changes introduced in it;
- Procedures of assuring the correspondence to standards in the branch of the software development, meeting which is defined in the assignment on specific software development;
- Algorithms of measurement and forming reports.

The software quality may be defined as correspondence to explicitly set functional and operational requirements, explicitly indicated standards of the development, and to implicit characteristics that are expected from professionally developed software. Such a definition of the software quality underlines three important circumstances:

- requirements to the software is a basis with respect to which the software quality is defined;



Jharko E..
Towards the Quality Evaluation of Software of Control Systems of Nuclear Power Plants: Theoretical Grounds, Main Trends and Problems.
DOI: 10.5220/0005534904710478
In *Proceedings of the 12th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2015), pages 471-478
ISBN: 978-989-758-123-6

- these standards define the set of criteria, which defines the software development style;
- there exists a manifold of implicit requirements, which are not frequently mentioned about (for instance, maintainability and updatability). If a software meets to explicit requirements to its development but is not in position to meet explicit requirements, then the software quality is doubtful.

These circumstances are most sharply traced with regarding software of highly reliable systems, to which, in particular, NPP APCS subsystems are related to, since besides complete correctness, the software possesses other characteristics being of interest to a consumer of this software, such as absence of errors under execution, integrity of data, time characteristics, accuracy, correctness of types, completeness, functional reliability, safety, maintainability, intelligibility, updatability, and others.

# 2 CLASSIFICATION OF SYSTEMS IMPORTANT FOR THE NUCLEAR POWER PLANT SAFETY

Under development of systems for power engineering, where the operation period of the main equipment is dozens of years, one should apply such solutions in the APCS, which would enable one to operate, repair, and update the installed equipment without stopping the technological process. Besides this requirement, providing high reliability, survivorship, and safety are the key requirements.

Analysis of advanced requirements, present status of hardware and software, tendencies of development enabled one to formulate a common approach to constructing systems for the power engineering: the systems are to be constructed either by use of own technologies, or by use of imported technologies. Meanwhile, the technologies imported are to be subject to an adaptation process that is to make them transparent and controllable to such a degree so as a supplier could expand his/her warranty obligations of duration of several dozens of years to them.

The series of IAEA standards on the safety (NS-R-1:2000) sets the notion on a classification of NPP systems in accordance to their importance for the safety. All devices, systems, and components, involving the software for monitoring and control, being safety important elements are defined and then classified on the basis on an implemented function and importance for the safety. These are designed, manufactured, and maintained in such a manner so as their quality and reliability would correspond to this classification.

The standard (IEC 61226 ed3.0) extends and specifies the IAEA classification, as well it sets criteria and ways that are to be applied under relating functions of monitoring and control of an NPP to one of the categories A, B, and C in dependence on the importance to the safety or to a not classified category for functions that do not play direct role in the safety assurance.

In accordance to the international classification (IEC 61226 ed3.0, 2009; Jharko, 2011), systems important for the NPP safe are separated from the point of view of functions implemented by these systems:

***Category A*** involves functions that play the main role in achieving or supporting the NPP safety in order to prevent development of emergencies to inadmissible consequences.

***Category B*** involves functions that play supplementary role with regard to the functions of category A in achieving or supporting the NPP safety, in particular the functions that are needed for operation under achieving a controlled status in order of preventing development of design events (DE) to inadmissible consequences or to moderate DE consequences.

***Category C*** involves functions that play an auxiliary or indirect role in achieving or supporting the NPP safety.

Table 1 present a comparison of safety classes of NPP systems presented in regulatory documents. In dependence on a safety class, software developed for these systems is imposed with limitations concerned with suitability of operation systems, programming languages, detail of documenting, etc.

The NPP APCS make-up involves systems of the 2nd, 3rd, and 4th safety classes in accordance to NP-001-97 or in accordance to the international classification (IEC 61226 ed3.0, 2009) (see Table 1) systems of classes A, B, C. Thus, under development of software for NPP APCS subsystems one should follow to standards of (IEC 60880 Ed. 2, 2006) (for systems of class A), (IEC 62138 Ed. 1, 2004.) (for systems of classes B, C).

Table 1: Comparison of safety classes of NPP systems.

<table>
<tr><th>Standard or regulatory document</th><th colspan="6">Safety classes (the importance degree increases from the left to the right)</th></tr>
<tr><td>NP-001-97</td><td colspan="2">Class 4</td><td colspan="2">Class 3</td><td>Class 2</td><td>Class 1</td></tr>
<tr><td rowspan="2">IAEA<br>NS-R-1:2000</td><td colspan="2" rowspan="2">Systems not important for the safety</td><td colspan="3">Systems important for the safety</td><td rowspan="2">No</td></tr>
<tr><td colspan="2">Systems concerned with the safety</td><td>Safety systems</td></tr>
<tr><td>IEC 61226:2009</td><td>Not classified</td><td colspan="2">Class C</td><td>Class B</td><td>Class A</td><td>No</td></tr>
<tr><td>IEEE 603:2009</td><td colspan="4">Not class 1E</td><td>Class 1E</td><td>No</td></tr>
</table>



# 3 THEORETICAL GROUND OF THE QUALITY EVALUATION ON THE QUALITY MODEL

The engineering of software development is applying the systemic approach, standards, and quantitative measurements of software characteristics to the design, operation, and maintenance of the software. Due to the development of technologies, the importance of software engineering persistently grows, and, correspondingly, methods of determining the software quality become increasingly called-for. The complexity of the process of development and maintenance of software is, due to many reasons, conditioned by special requirements to its quality. This factor justifies the importance of development of formalized methods of control the software quality. At present, several definitions of the notion of the software quality are used, which are, generically, are compatible with each other. Generalizing definitions of standards, one may conclude that the software quality is an ability of a software product to meet set or assumed demands under operation within given conditions.

The software quality plays an important role for all system as a whole. So, the software quality is considered as a very important aspect for developers, users, and managers of projects. The software quality is a quantity reflecting the volume of involvement of the software product into a set of desired functions to increase the software product efficiency in course of the life cycle (Firesmith, 2003). For any software using system three types specifications are to be developed, such as functional requirements, requirements to the quality, requirements to resources. The quality involves all characteristics and essential features of a product or its performance that are related to meeting requirements set by specifications.

The quality is generalization of characteristics or features of a product or works, which are related to the ability of a software product developed to meet requirements set. The software quality may be separated on two constituent parts, such as the quality of software development and the quality of software product. The software development connecting such elements as technologies, tools, employees, organization, and equipment is considered in the context of the quality of procedures of software development. Nevertheless, the software product quality consists of certain aspects, such as clarity of documentation and integrity, project traceability, software reliability, and completeness of testing main characteristics of the software product. A software model is usually defined by set of characteristics and relations between them, which actually provide a basis for both defining quality requirements and evaluation of the software quality (ISO/IEC 9126-1, 2001; GOST 28195-89). The quality model may also be defined as a structured set of properties that are required to meet certain purposes (Fitzpatrick, 1996). An advantage of the quality model is a decomposition of significant for software objects, such as life cycle, software quality, on a set its characteristics/subcharacteristics. The quality, besides a description and measurement of functional aspects of software, also describes additional functional properties, such as "how this product was created" and "how it performs".

Software users need to create models of the software quality to evaluate the quality both qualitative and quantitatively (Jharko, 2014). Quality models that are available at present are in majority of cases hierarchical models based on quality criteria and indexes (metrics) concerned with it. All quality characteristics may be separated on three categories in accordance to methods, which basis they were created on. To the first class, theoretical models may be related, based on a hypothesis of relations between software variables. To the second class, models of "data control" are related, based on a statistical analysis. And, finally, a combined model, in which the intuition of a researcher is used to determine a required type of the model, while the data analysis is used to determine quality model constants. But all these model associate user's interests, that is outgoing (output) properties of the system with internal properties that are understandable to developers.

In the ground of quality models, a multi-level approach lies (the number of layers may be 2 /models of Mc Call and Boehm/ or 3 layer /involving metrics/) (see Fig. 1), that is quality characteristics are separated on three groups:

- ***factors***, describing a software product from the point of view of user and set as requirements;
- ***criteria***, describing software product attributes from positions of a developer and set as purposes;
- ***metrics***, used for quantitative measurement of availability of a factor in the system.

A comparative analysis of software quality models is presented in Table 2.



Table 2: Comparative analysis of software quality models.

| Quality Characteristics | Models | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | **Mc Call** (Mc Call et al., 1997 –c) | **Boehm** (Boehm et al., 1978) | **FURPS/ FURPS+** (Grady, et al. 1987) | **Ghezzi** (Ghezzi et al., 1991) | **Dromey** (Dromey, 1995) | **ISO 9126** | **Kazman** (Bass at al. 2003) | **Хосрави** (Chang, 2008) | **Sharma** (Sharma, 2008) |
| **Accuracy** | | | | + | | | | | |
| **Availability/Reliability** | + | + | + | + | + | + | + | | + |
| **Correctness** | + | | | | | | | | |
| **Efficiency** | + | + | + | + | + | + | + | | + |
| **Flexibility** | + | | | + | | | + | + | |
| **Functionality** | | | + | | + | + | + | | + |
| **Human Engineering** | | + | | | | | | | |
| **Integrity** | | | | + | | | | | |
| **Interoperability** | + | | | | | | | | |
| **Maintainability** | + | + | + | + | + | + | + | | + |
| **Modifiability** | | + | | | | | | | |
| **Performance** | | | + | | | | | | |
| **Portability** | + | + | | + | + | + | | | + |
| **Process Maturity** | | | | | + | | | | |
| **Reusability** | + | | | + | | | | + | |
| **Robustness** | | | | | | | | + | |
| **Scalability** | | | | | | | | + | |
| **Security** | | | + | | | | + | | |
| **Supportability** | | | + | | | | | | |
| **Testability** | + | + | | | | | + | | |
| **Understandability** | | + | + | | | | | | |
| **Usability** | + | | + | + | + | + | + | + | + |

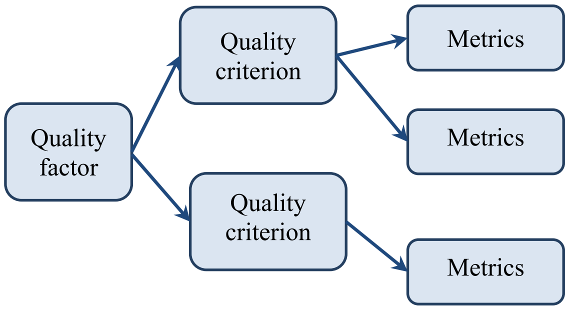


Figure 1: Quality model structure.

# 4 SOFTWARE QUALITY ASSURANCE

Software provides considerable impact in functions implemented by safety important systems. Software may support additional functions introduced in accordance to the design of a developed or already operated system (for instance, initialization and monitoring of hardware, communication between subsystems, etc.). For NPP safety important systems, the life cycle of software is closely concerned with the life cycle of the safety of the system itself, as well the specification of requirements to the software is a part the system specification. Any violations in the technological process of software development may lead to undesirable results:

- cost rise of the software product due to increasing time of its development;
- due to errors not revealed under testing:
  - as a minimum, this leads to decreasing the software product capacity,
  - as a maximum, this leads to decreasing of the safety of safety critical systems;
- errors, unclear messages, unfriendly interface, careless documenting create inconvenience for users, what leads them to selecting more qualitative software product of a competitor.

The software quality is hard to achieve, since the process of obtaining the required software quality touches the process of development, methods and control of the process. The software quality is achieved due to applying the methodology of development and using methods of verification and validation in course of the life cycle of the software development for NPP safety important systems. In Fig. 2, the place of the software verification and validation is presented in the context of the quality assurance and the hierarchy of standards in the branch of development software for NPP safety important systems. Fig. 3 presents a practical illustration of the V-shape software development most frequently used for software of NPP safety important systems. The left side of the V-shape scheme is the design and verification, while the right one is the implementation and validation of the software design.

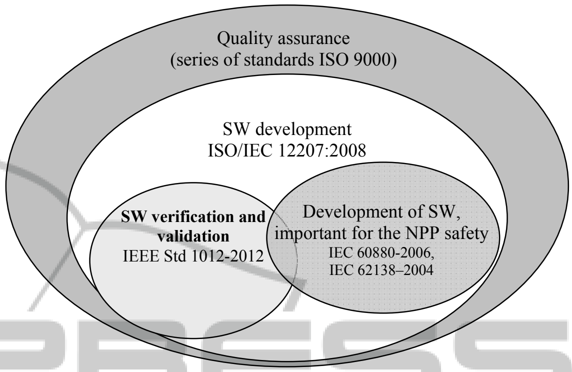


Figure 2: The place of V&V of the software in the quality assurance of software important for the NPP safety.

At the present stage of the scientific experience in order to improve the quality of developed software, there were, are elaborated and improved standards enabling one, from one hand side, to provide the transparency of procedures of the software quality assurance, and, from another hand side, to achieve the software quality due to description detailing.

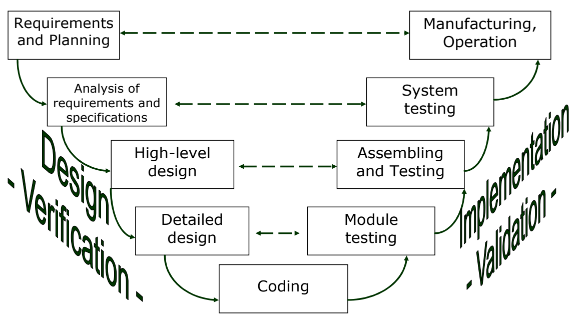


Figure 3: V-shape model of the life cycle of software to assure the software quality of NPP safety important systems.

# 5 SOFTWARE QUALITY EVALUATION

The software quality is defined in the standards (ISO/IEC 9126-1:2001) and (ISO/IEC 25010:2011) as any totality of software characteristcs relating to a possibility to meet expressed or assumed demands of all interested parties.

There are distinguished the notions of the internal quality concerned with software characteristics as itself, disregarding its behavior; external quality characterizing the software from the point of view its behavior; and software quality under use in different contexts, that quality, which is perceived by users under specific scenarios of software performance. For all these aspects of the quality metrics were introduced, enabling one to evaluate the quality. Besides that, to create a reliable software the quality of technological processes of its development is essential. The interrelations between these aspects of the quality in accordance to the scheme adopted by ISO/IEC 9126 (ISO/IEC 9126-1:2001; ISO/IEC TR 9126-2:2003, ISO/IEC TR 9126-3:2003, ISO/IEC TR 9126-4:2004) is presented in Fig. 4.

Table 3 presents the order of the software evaluation. The software quality may be considered as "sufficiently good", when potentially-positive results of creating and operating the software acceptably overbalance potentially-negative opinions of customers. Such an approach checks from the point of view of the conventional notion of the software quality different variants of the implementation. Under such an approach to the software quality, high unchecked requirements are substituted with optimal ones. This approach is focused on identifying tasks and improving possibilities for decision making. Thus, the process of software development for NPP safety important systems is to be sooner problem-oriented rather than purpose-directed to the software quality. Also one may say that the software quality, in accordance to the notion of "sufficiently good" is the optimal set of solutions of this totality of tasks. Such a way of the interpretation is to coordinate the considered tasks, elaborate compromise variants, confronting them with corresponding processes of the life cycle (ISO/IEC 12207:2008).

As pointed above, the quality is a "totality of object characteristics that have a relation to its ability to satisfy to set and assumed demands". By use of the term "satisfaction", the ISO/IEC 9126 standard assumes the "possibilities of software for satisfaction of users in the ser context of use". In Fig. 5, factors and attributes of the internal external software quality are presented in accordance to ISO/IEC 9126.

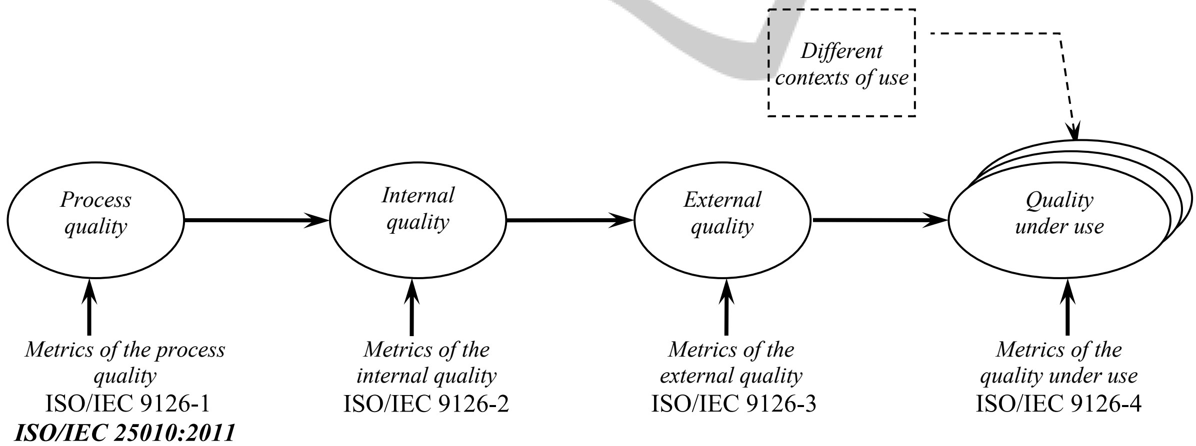


Figure 4: Basic aspects of the software quality in accordance to standards of ISO/IEC 9126-1:2001 and ISO/IEC 25010:2011.

Table 3: The order of evaluation of the quality of software products.

| **Parties interested in quality evaluation** | **Stages of the life cycle of a software** | | | | | |
|---|---|---|---|---|---|---|
| | **Development** | **Tests** | **Replication** | **Adoption** | **Maintenance** | **Operation** |
| **Developer** | Yes | Yes | Yes | Yes | Yes | Yes |
| **Test and certification centers** | - | Yes | - | Yes | - | Yes |
| **User** | - | - | - | - | - | Yes |

However software quality standards available at present do not touch completely issues of the information security and cybersecurity, which have become vital in the last years for software of safety important systems. A main direction in the assurance of the software quality of safety important systems is to become observing the requirements of the IEC 62645-2014 standard.

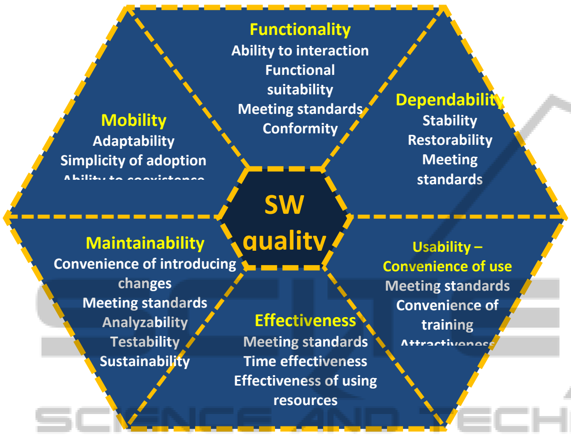


Figure 5: Factors and attributes of the external and internal software quality in accordance to ISO/IEC 9126.

## REFERENCES

Bass, L., Clements, P., and R. Kazman, 2003. *Software Architecture in Practice*, 2nd ed., Addison Wesley, 2003, 528 p.

Boehm, B.W., Brown, J.R., Kaspar, H., Lipow, M., MacLeod, G.J., and M.J. Merritt, 1978. *Characteristics of Software Quality*, TRW Series of Software Technology, North Holland, Amsterdam, 1978, 166 p.

Byvaikov, M.E., Zharko, E.F., Mengazetdinov, N.E., Poletykin, A.G., Prangishvili, I.V., and V.G. Promyslov, 2006. “Experience from design and application of the top-level system of the process control system of nuclear power-plant”. *Automation and Remote Control*, vol. 67, no. 5, pp. 735-747.

Chang, C., Wu, C., and H. Lin, 2008. “Integrating Fuzzy Theory and Hierarchy Concepts to Evaluate Software Quality”, *Software Quality Control*, vol. 16, no. 2, pp. 263-267.

Dromey, G.R., 1995. “A model for software product quality”, *Transactions of Software Engineering*, vol. 21, no. 2, pp. 146-162.

Fitzpatrick, R., 1996. *Software Quality: Definitions and Strategic Issues*, Staffordshire University, School of Computing Report, 1996. 34 p.

Firesmith, D.G., 2003. *Common concepts underlying safety, security, and survivability engineering*, Technical Note CMU/SEI-2003-TN-033, Carnegie Mellon Software Engineering Institute.

IEC 60880 Ed. 2, 2006. Nuclear power plants – Instrumentation and control systems important to safety. Software aspects for computer-based system performing category A function. 2006.

IEC 61226 ed3.0, 2009. Nuclear power plants – Instrumentation and control important to safety – Classification of instrumentation and control functions. 2009.

IEC 62138 Ed. 1, 2004. Nuclear Power Plants – Instrumentation and Control Computer-based systems important for safety. Software for I&C systems supporting category B and C functions. 2004.

IEC 62645 ed1.0, 2014. Nuclear power plants - Instrumentation and control systems - Requirements for security programmes for computer-based systems.

IEEE Std 1012-2012, 2012. IEEE Standard for System and Software Verification and Validation.

IEEE Std. 603-2009, 2009. IEEE Standard Criteria for Safety Systems for Nuclear Power Generating Stations.

ISO/IEC 12207: 2008. Systems and software engineering – Software life cycle processes.

ISO/IEC 9126-1: 2001. Software engineering – Software product quality – Part 1: Quality model.

ISO/IEC TR 9126-2: 2003. Software engineering – Product quality – Part 2: External metrics.

ISO/IEC TR 9126-3: 2003. Software engineering – Product quality – Part 3: Internal metrics.

ISO/IEC TR 9126-4: 2004. Software engineering – Product quality – Part 4: Quality in use metrics.

Jharko, E.Ph., 2008. “Design of Intelligent Information Support Systems for Human-Operators of Complex Plants”, *IFAC-PapersOnLine. ISSN: 1474-6670*, vol. 17, part 1, World Congress, pp. 2162-2167.

Jharko, E.Ph., 2011. “Assessment of the software quality of systems important for the NPP safety”, *Information Technologies and Computing Systems*, no. 3, pp. 38-44. (in Russian).

Jharko, E.Ph., 2014. “Comparison of software quality models: the analytical approach”, *XII All-Russian Control Congress VSPU-2014. Moscow, June 16-19, 2014.: Proceedings*, V.A. Trapeznikov Institute of Control Sciences, Moscow, 2014, pp. 4585-4594. (in Russian).

Jharko, E.Ph., and O. Zaikin, 2011. “The Flexible Modeling Complex for an NPP Operator Support System”, *IFAC-PapersOnLine. ISSN: 1474-6670*, vol. 18, part 1, World Congress, pp. 12156-12161.

Ghezzi, C., Jazayeri, M., and D. Mandrioli, 1991. *Fundamental of Software Engineering*, Prentice-Hall, 1991.

GOST 28195-89, 1989. Quality control of software systems. General principles. (in Russian).

Grady, R.B. and D.L. Caswell, 1987. *Software Metrics: Establishing a Company-Wide Program*, Prentice-Hall, 1987, 275 p.

McCall, J.A., Richards P.K., and G.F. Walters, 1977a. *Factors in Software Quality: Concept and Definitions of Soft-ware Quality*, Final Technical Report, vol. 1, National Technical Information Service, Springfield, 1977.

McCall, J.A., Richards, P.K., and G.F. Walters, 1977b. *Factors in Software Quality: Metric Data Collection and Validation*, Final Technical Report, vol. 2, National Technical Information Service, Springfield.

McCall, J.A., Richards, P.K., and G.F. Walters, 1977c. *Factors in Software Quality: Preliminary Handbook on Software Quality for an Acquisition Manager*, Final Technical Report, vol. 3, National Technical Information Service, Springfield.

NP-001-97 (PNAE G-01-011-97), 1997. General guidelines on safety assurance of nuclear power plants. OPB-88/97, Gosatomnadzor Rossii, Moscow.

NS-R-1: 2000. Safety of Nuclear Power Plants: Design Safety Requirements. IAEA Safety Standards Series No. NS-R-1.

Poletykin, A.G., Jharko, E.Ph., Zuenkova, I.N., Promyslov, V.G., Byvaikov, M.E., and N.E. Mengazetdinov, 2006. "Software for nuclear power engineering", *Automation in Industry*, no. 8, pp. 52-56. (in Russian).

Sharma, A., Kumar, R., and P.S. Grover, 2008. "Estimation of Quality for software components: an empirical approach", *ACM SIGSOFT Software Engineering Notes*, vol. 33, no. 6, pp. 1-10.